編集の際に便利なgitの使い方

この記事をgithubのページ

(https://github.com/KatsuyaITO/episode/blob/master/MAY2017/gitadv.pdf)
で開いている場合は右上の[Download]ボタンをクリックして,
ファイルを直接ダウンロードしてください.

これからコマンドラインでgit を操作する際に,できるだけ覚えておいて欲しい,
覚えておくと便利なコマンドを代表的な使い方とともに列挙します.
そのコマンドでググるなりして,理解して使ってください.

1.git clone (自分の環境にファイルをダウンロード)

自分の好きなところにgithubレポジトリーをダウンロードすることができる.
新たな環境を作るときに便利.
git clone https://github.com/KatsuyaITO/episode.git
などとする.

2.git status (今の状態を把握する)

git cloneしてローカルで編集をした後はどのファイルを編集したのかをみることができます.
git statusでどのファイルを編集したのかを確認し,
git add で新しいファイルをコミットに追加し,
git checkout でその変更をコミットしないようにできます.

3.git commit (変更を保存する)

記事を変更した際に,それを保存することを**commitする**といいます.
githubのwebページ上で編集した場合が自動的にcommitを作ってくれます.
コマンドライン上でやる場合には
git commit -v -a -m “コミットメッセージ”
などとするとできます.

ローカルでcommitしただけでは,その変更はgithub上には上がらないので,
git pull —rebase
git push origin (branch名)

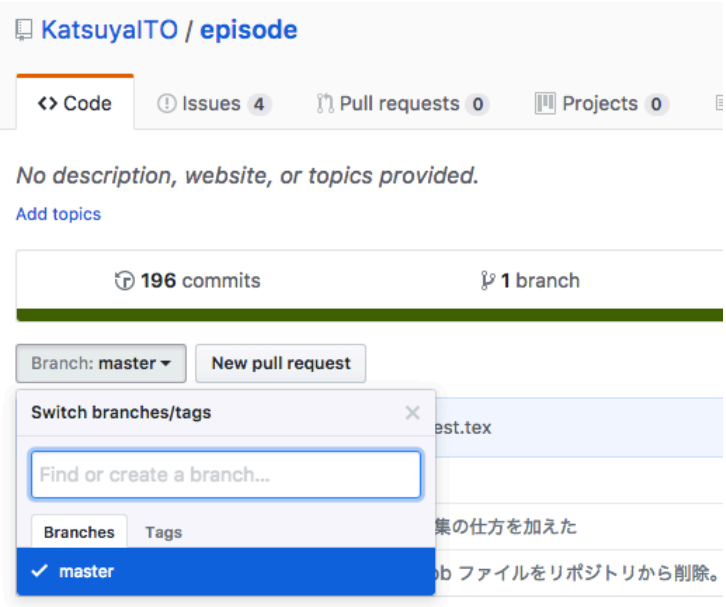
などとする.

4.branch -> pull request (大きな編集する際に提案する)

ある人が作った文章に対して「こことここととここと」を編集したいなあと思うことが有ります.
そのときに軽微な編集ならば直接編集してしまっても良いのですが,
一度別の場所で編集をして,このように編集してはどうですか,と提案することができます.

一度別の場所で編集できるようにすることを**branchを切る**と言って
このように編集してはどうですかと提案することを**pull requestを送る** といいます.

branchを切る 操作は以下の Branch:masterボタンを押すとできます.



pull requestを送るは

githubのページの上側にあるPull requestsタブをクリックして,
New Pull Requestsをクリックするとできます.

4.git rebase -i (歴史を改ざんする)

たくさんコミットをしてしまって,コミットを一つにまとめたいときは,

git rebase -i HEAD~~~
~の個数は変更するコミットの個数
などとコマンドラインで操作するといいです.(詳しくはググって)
歴史をした際には git push -f という強制pushをしないとレポジトリーに変更は知らせられない.

5.tig (歴史を見る)

これまでのコミットの履歴がtigコマンドで見ることができます

6.git reset —hard (タイムスリップする)

git reset —hardコマンドを使うと好きな時点まで戻ることができます.
その際に変更履歴はgit reflog で見ると便利です.